Putting ROSE To Work:
A Proposed Application of a Request-Oriented Scheduling Engine for Space Station Operations

Presented to SpaceOps 2000
June 19th – 23rd, 2000
By

John Jaap
John.Jaap@msfc.nasa.gov
256-544-2226

Kim Muery
Kim.Muery@msfc.nasa.gov
256-544-3949

Mission Support Systems
Flight Project Directorate
Marshall Space Flight Center
National Aeronautics and Space Administration

Abstract

Scheduling engines are found at the core of software systems that plan and schedule activities and resources. A Request-Oriented Scheduling Engine (ROSE) is one that processes a single request (adding a task to a timeline) and then waits for another request. For the International Space Station, a robust ROSE-based system would support multiple, simultaneous users, each formulating requests (defining scheduling requirements), submitting these requests via the internet to a single scheduling engine operating on a single timeline, and immediately viewing the resulting timeline. ROSE is significantly different from the engine currently used to schedule Space Station operations. The current engine supports essentially one person at a time, with a pre-defined set of requirements from many payloads, working in either a "batch" scheduling mode or an interactive/manual scheduling mode. A planning and scheduling process that takes advantage of the features of ROSE could produce greater customer satisfaction at reduced cost and reduced flow time. This paper describes a possible ROSE-based scheduling process and identifies the additional software component required to support it. Resulting changes to the management and control of the process are also discussed.

Introduction

Scheduling payload operations has historically been done by a cadre of mission planners who act as agents for the payload developers. The members of this cadre are experts in the features, capabilities, limitations, and language of the scheduling engine. Different members of the cadre have different scientific and technical backgrounds; they are usually matched to the payload they represent. The members become experts in the requirements of the payload that they represent – often with the help of extended input from the payload developers.

The process of collecting requirements is often protracted, with multiple iterations. The payload developer submits payload requirements and descriptions in the requested format. A cadre member reviews the information and contacts the payload developer for a better understanding. The payload developer modifies his submission.

After the cadre has an adequate (in-depth) understanding of the requirements, the cadre members use the scheduler to produce a timeline. The payload developers review the timeline, request changes, and the cadre makes repairs to the timeline. Usually, several months elapse between the requirements submittal and the completion and publication of the timeline.

The scheduler currently used for the International Space Station payloads requires expertise to represent the payload requirements. The lack of several key features causes the cadre to spend a great deal of time describing a payload's actual scheduling requirements in the "vernacular" of the scheduler, and then to hand build the timeline with a manual timeline editor to get the desired results. The scheduler cannot represent or process optional sequences of operations (multiple scenarios), choice of constraints within a non-homogeneous group, or soft (fuzzy) requirements. Furthermore, the current scheduler only supports one cadre member at a time, so the cadre members must either take turns or funnel all scheduling through one member.

A scheduling system that allowed payload developers to schedule their own payloads would produce a better timeline (one that met the payload developer's requirements) in a shorter time period. The delay between describing and submitting a scheduling request and viewing the results would be reduced to minutes rather than months. The payload developers could then refine and resubmit their requirements until they got a first-rate timeline. The cadre would only have to preload the scheduling engine with the availabilities and allocation constraints, define the station and payload equipment resource requirements, and post-check that the timeline is safe and meets programmatic constraints.

Two obstacles must be overcome to deploy such a system:

1. The software system itself must be built.
2. An operations concept using the system must be designed and accepted.

The Software System

The Mission Support Systems Group at MSFC has embarked on an effort to design and demonstrate a software/hardware system that allows the payload developers to schedule their own payload activities (Jaap, Davis, 2000). It includes a graphics-based method for formulating scheduling requests and a centrally located, multi-user Request-Oriented Scheduling Engine (ROSE) working against one current set of resource availabilities and one current timeline. The system uses the World Wide Web to allow the user community (the payload developers – the customers) to readily access the system from a personal computer or workstation. The name ROSE is being applied to both the project as a whole and to the scheduling engine itself.

ROSE is a web-based application. The users access the system via the World Wide Web; no ROSE-specific software is installed on a user's computer and no data is stored on a user's computer. The user navigates via a web browser to the ROSE web site, logs on, and proceeds to formulate scheduling requests and submit them for scheduling. The user can also review the in-work timeline, delete tasks from the timeline, and view ancillary information such as resource availabilities and allocations. An overview of the ROSE architecture is shown in Figure 1.

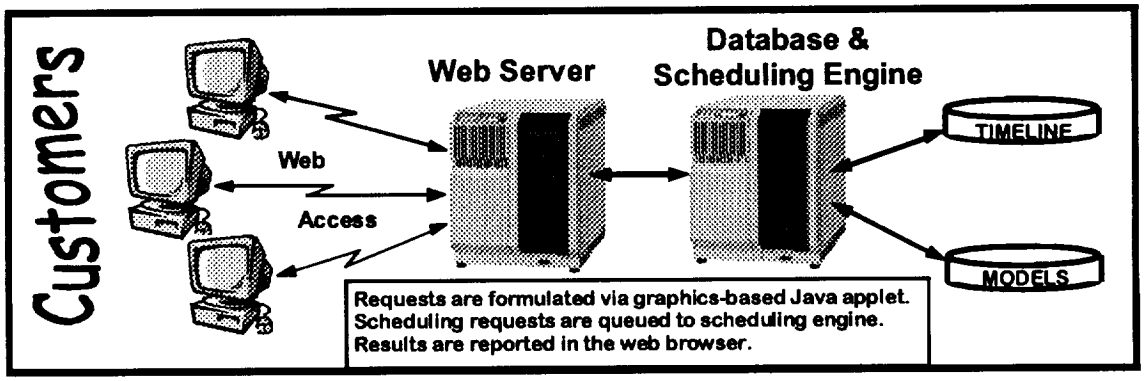

Figure 1. ROSE Architecture Overview

Maintenance of a web-based system is literally zero. Opening the web page with a browser will download and execute the latest version of the client without any effort on the user's part. The user can switch between client computers in his office, home, or even use a portable without installing ROSE-specific software or moving any files. Security is the only concern, and security is being integrated into the ROSE research and development effort so that an adequate solution can be deployed. The architecture of the ROSE system is further described in the reference (Jaap, Davis, 2000).

The Critical Element Modeling in the context of activity scheduling has historically meant defining an activity's requirements for shared resources (power, crew, etc.), time-dependent constraints (when the vehicle is within view of a target), and sequencing of activities relative to each other (warm-up before data-take before shutdown). In the context of ROSE, modeling is also called "request formulation," because a model is the core of a scheduling request.

Modeling is the process of representing the requirements in a manner usable by the scheduling engine so that it can produce a correct and acceptable timeline. Modeling always requires an in-depth

understanding of the payload and how the scheduling engine behaves. In the current operations concept, a payload developer provides an in-depth description of the payload to the group, called the scheduling cadre, that eventually runs the scheduler to produce the timeline. After the timeline is completed and published, the payload developers review the timeline and write execution change requests. The execution team makes the changes to the timeline.

If payload developers, who already have expertise in the payloads, are to formulate scheduling requests and submit them to a scheduling engine, they must have expertise in the scheduling engine – this is critical. Being an expert in the behavior of the scheduling engine means knowing how the engine will react to a given model, and how to build a model to achieve the desired results. ROSE will make the payload developers virtual experts in using the scheduling engine by making them experts in modeling. ROSE uses a request format that is a natural representation of the requirements without adding artificial constraints or constructs. We call this modeling methodology high-fidelity or hi-fi modeling – the model looks like the real world payload and the engine interprets the model as expected. ROSE provides immediate feedback (when a request is submitted, the resulting timeline is available immediately), thereby exposing users to the workings of the scheduling engine. After only a few submittals, the users will know what to expect from the engine when a request is submitted; i.e., they will become virtual experts.

In addition to supporting "hi-fi" models, ROSE provides a user-friendly interface that is neither laborious nor time consuming and which requires little or no training. The user interface of ROSE is the same as that of the Interim User's Requirements Collection (iURC) system currently being used to collect payload scheduling requirements for at least the first four increments of the International Space Station. This interface is described in a previous paper (Jaap, Davis & Meyer, 1997) and in the on-line user's manual for iURC (http://payloads.msfc.nasa.gov/iURC/help. The modeling process itself employs graphical methods to describe activities and sequences.

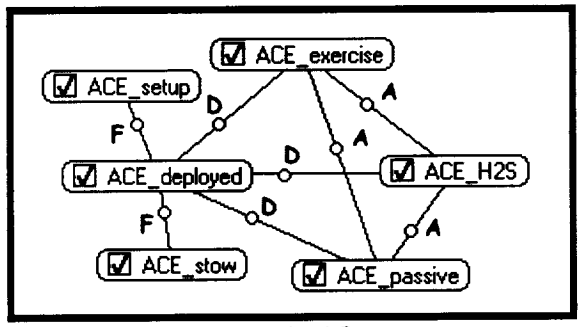

Figure 2. Typical Sequence

The user is presented with a canvas to which items are added and arranged in a hierarchy (for activities) or in a network (for sequences). Details of the requirements are entered via dialog boxes. The advantages of using a graphics method are best illustrated by example. In the typical International Space Station sequence shown in Figure 2, the temporal relationships marked with an F are "followed by" relationships, those marked with a D are "during," and those marked with an A are "avoid." The sequence indicates that ACE_setup is followed by ACE_deployed, which is followed by ACE_stow; during ACE_deployed, ACE_H2S, ACE_passive, and ACE_exercise are done, but while avoiding one another. ACE_H2S, ACE_passive, and ACE_exercise are themselves sequences. This example is a simple sequence; International Space Station users frequently submit sequences with twenty or more tasks and many relationships, including some to station tasks like reboost and shuttle_docking. The graphics representation is understandable even when extended to large and complex sequences. The modeling methodology supports optional arrangements of tasks within the sequence (sequence scenarios), soft requirements, and choice of constraints within a group with lock-in across sequence members.

There is no explicit configuration control of the user's models. A user can always edit his models. However, when a model is added to the timeline (scheduled), an abstract of that model is saved with the timeline and never edited. Thus, editing a model does not edit what is already scheduled, and configuration control of users' models is not required. The modeling methodology of ROSE is further described in the reference (Jaap, Davis, & Meyer, 1997).

Scheduling While the scheduling engine is the key element of the system, the user should not need to know how it works – the user needs only to hone his modeling skills. The scheduling engine must understand the "language" of modeling perfectly and should behave exactly as expected. Moreover, the scheduling engine must provide feedback (reports for successes and explanation for

failures) to help the user improve his modeling skills. To be useful, the ROSE system as a whole must respond to scheduling requests from each of its multiple users in only a few minutes.

In ROSE, each scheduling request initiates an attempt to schedule one performance of one sequence that may have embedded sub-sequences, repeated tasks, and multiple scenarios. The scheduling request is primarily the data in the model, but the user can override or add additional constraints such as a scheduling window, starting time frame, and scenario specification.

Due to the nature of ROSE, some characteristics of the scheduling engine are compulsory. It must handle the models. It must respond quickly. Everything it schedules must be "valid;" i.e., resources are never oversubscribed and constraints are never violated. Once something is in the timeline, only the user can remove it; the scheduling engine doesn't adjust one task to fit another task into the timeline. It must constrain each account so that its resource allocations are not exceeded. Furthermore, ROSE must not lose anyone's scheduled tasks if the scheduling engine or the computer crashes – scheduled tasks are committed to the database immediately after scheduling.

The Operations Concept

In addition to designing and demonstrating a software system which allows payload developers to schedule their own payloads, the Mission Support Systems Group is also proposing several ROSE-based operations concepts for payload planning and scheduling for the International Space Station (ISS) – one of which is described in this paper. This proposal is based on an instance of ROSE installed at a major support facility, such as the U.S. Payload Control Center, and serving all of the payload developers supported by that facility. The payload developers would remotely connect to the ROSE system and simultaneously schedule their payloads. The results would be the timeline provided by that installation (or partner) to the Payload Operations Integration Function.

Comparison of Concepts

An overview-level comparison of the proposed concept with the current operations concept can provide a clearer picture of the ROSE-based concept. While the processes (that implement the concepts) have only minor timing differences, the proposed ROSE-based process results in the payload developers, not the cadre, creating the schedule. The proposed concept creates different, but less demanding, tasks for the cadre. The proposed concept greatly improves the quality of the schedule before it goes to the execution phase and therefore significantly reduces the number of change requests that must be written, processed, and implemented. Both the current concept (as applied to US payloads) and the proposed concept have a preparation phase during which planning requests are submitted to the cadre by the payload developers and equipment definitions are created. Both concepts have a weekly scheduling phase that produces the timeline to be executed during the second week after it is produced. During any week, three actions are occurring: the week after next is being scheduled, next week is being prepared and up-linked, and the current week is being executed. The preparation phase is closely linked to what equipment is on board, which is linked to crew change-out or the arrival of a re-supply ship. In ISS nomenclature an "increment" is a period of time that is punctuated by a crew change-out or a re-supply visit; nominally, an increment is 90 days. The preparation phase for an increment precedes the start of the increment; the scheduling phase begins two weeks before the increment starts and continues for the duration of the increment.

The Current Process The preparation phase starts when the payload developers submit their requirements to the scheduling cadre, usually several months before the start of an increment. Collecting requirements is often protracted because multiple iterations are required. The payload developer submits payload requirements and descriptions in the requested format. A cadre member reviews the information and contacts the payload developer for a better, often in-depth, understanding of the requirements. The payload developer modifies his submission. Because the payload developer does not have access to the scheduling engine, is not familiar with its limitations, and does not know how it will handle a request, the payload developer does not have and cannot develop the expertise required to submit "good" requests that will go directly into the schedule without the "hand-holding" of a cadre member. Toward the end of the preparation phase, the models are placed under configuration control and can only be changed via a change-request process.

Also, during the preparation phase, the cadre members build equipment models which define how much of each station resource is used when the equipment is operated. Most equipment has multiple operation modes with differing resource requirements; each of these modes is modeled. The equipment models are built and edited by the cadre with payload developer input and approval. Equipment models include both information received from the payload developers and requirements derived from how the equipment is connected (integrated) with the ISS. Also, during the preparation phase, the cadre generates a rough plan for the increment.

During the scheduling phase, the cadre members update the equipment models to reflect the current situation on the ISS and late requests from the payload developers. The payload developers submit change request against the models that were written in preparation phase. Then the cadre members transfer the models to the scheduling program, and, using the scheduling program, produce a timeline. Producing a timeline often requires minor and sometimes major changes to the models; i.e., scheduling includes an iteration loop where knowledge gained from scheduling or attempting to schedule a model is used to adjust the model to produce the desired timeline. Due to the shortness of the scheduling phase, it is sometimes necessary to make these changes without consulting with the payload developers.

After the cadre has completed the scheduling process, the timeline is delivered to the integration function where it is integrated with timelines from other ISS partners. Simultaneously, it is "published" so that the payload developers can review the timeline. Usually, several weeks or months have elapsed since requirements submittal. If a payload developer wants to have the timeline changed, an execution change request is written and submitted to the execution team. An overview of the current process is shown in figure 3.

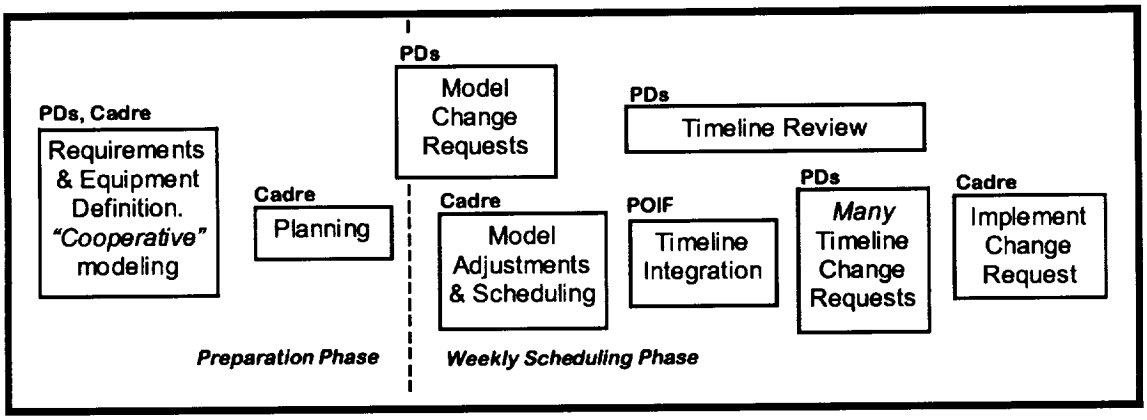

Figure 3. Overview of Current Scheduling Process

The ROSE-based Process Again, the preparation phase starts when the payload developers submit their requirements to the scheduling cadre, usually several months before the start of an increment. However, collecting requirements is not as tedious as with the current process for three reasons: the ROSE scheduling engine can schedule more complex models, the payload developers get feedback from the scheduling engine (during the scheduling phase), and the cadre has no need to have an in-depth understanding of the payload. In the ROSE-based concept, the payload developers create the actual models to be scheduled and will fine-tune the models during the scheduling phase. Nevertheless, cadre members review the models and make suggestions. Cadre members also create reference models that will be compared to the scheduled models during the timeline validation task; these are available to the payload developers.

In the ROSE-based concept, modeling of equipment modes is identical to the current concept. Likewise, the cadre produces the same rough plan for the increment.

As in the current concept, during the scheduling phase, the cadre members update the equipment models to reflect the current situation on the ISS and late requests from the payload developers.

(Here the proposed concept differs greatly.) Based on the increment plan produced during the preparation phase, and other information, the cadre would generate daily allocations per payload for the week to be scheduled. The allocations are not profiles but are total usage limits of each resource during the planning week. Once the system is initialized with all the resource constraints, the *payload*

developers use the ROSE system to produce a timeline. As always, producing a good timeline requires attempting to schedule a model, rejecting unacceptable results, tweaking the models, and trying again. The ROSE-based concept places the payload developer in the middle of this important iteration loop. No one knows the payload requirements or what is desired in the timeline better than the payload developer, and no one can produce a timeline as good as the one the payload developer can produce.

After the payload developer has completed the scheduling process, the timeline is delivered to the cadre who verifies that the models are valid. That is, they verify that the payload developer did not "fudge" to get the models into the timeline. The ROSE-based concept will show the payload developer where a small change in the requirements would allow a model to be scheduled; there will be a temptation to make the small changes even when this means "under-stating" the requirements. The payload developer cannot change the resources used by the equipment models, but the duration of the equipment usage or the mode of the equipment could be changed so that requirements are "under-stated." The cadre will use a software module that compares the scheduled models to the reference models (generated during the preparation phase) to verify that the scheduled models are legal. The cadre will also visually inspect the models and timeline.

After the timeline is verified, it is passed to the integration function where it is integrated with timelines from other ISS partners. Simultaneously, it is "published" so that the payload developers can review the timeline. If a payload developer wants to have the schedule changed, an execution change request is written and submitted to the execution team. Since the payload developer just produced the schedule (of his payload), it is unlikely that changes will be required. An overview of the ROSE-based process is shown in figure 4.

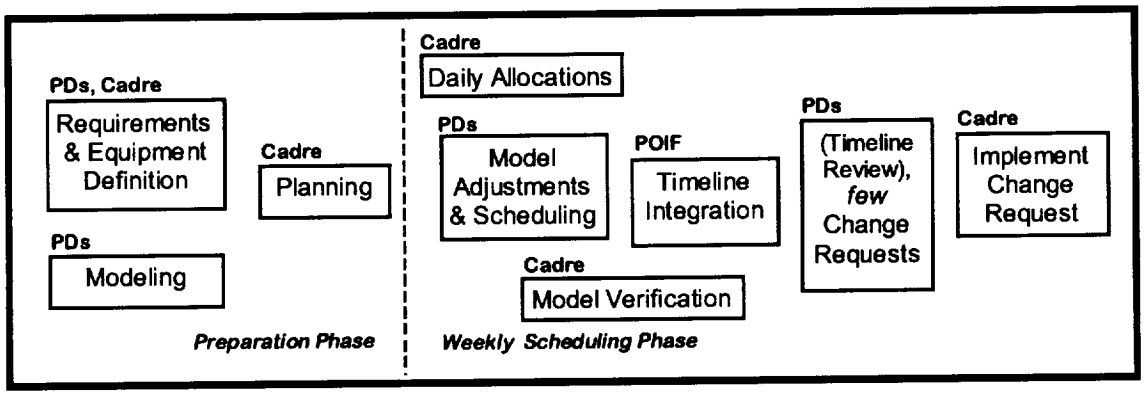

Figure 4. Overview of ROSE-based Scheduling Process

Details of the ROSE-based Process

The flow diagram shown in figure 5 on the next page shows the principal information flow of a ROSE-based process. The tasks done by the payload developers are shown on the left side, and those done by the cadre are shown on the right side.

Preparation Phase The process starts several weeks before the start of an increment when the payload developers submit their planning requests and their payload equipment definitions. Meanwhile the cadre is defining the resources to be available during the increment and reviewing the applicable Increment Requirements Definitions Document (IDRDs). Based on the equipment definitions from the payload developer, the resource list, and the expected on-board configurations, the cadre creates the models of the equipment. For each mode in which the equipment may operate, the equipment models define what resources (except crew) and conditions are used in what amount. Where a choice of resources is to be made by the scheduler, those choices are also defined. An equipment model may reference another equipment model so that a hierarchy of requirements can be modeled. For more on modeling implicit requirements see the reference (Hagopian, Maxwell, 1996).

The cadre also develops an increment plan based on the payload developer's requests, the predicted resource availabilities, and the IDRDs. This plan will be used later to produce the daily allocations of resources for each payload.

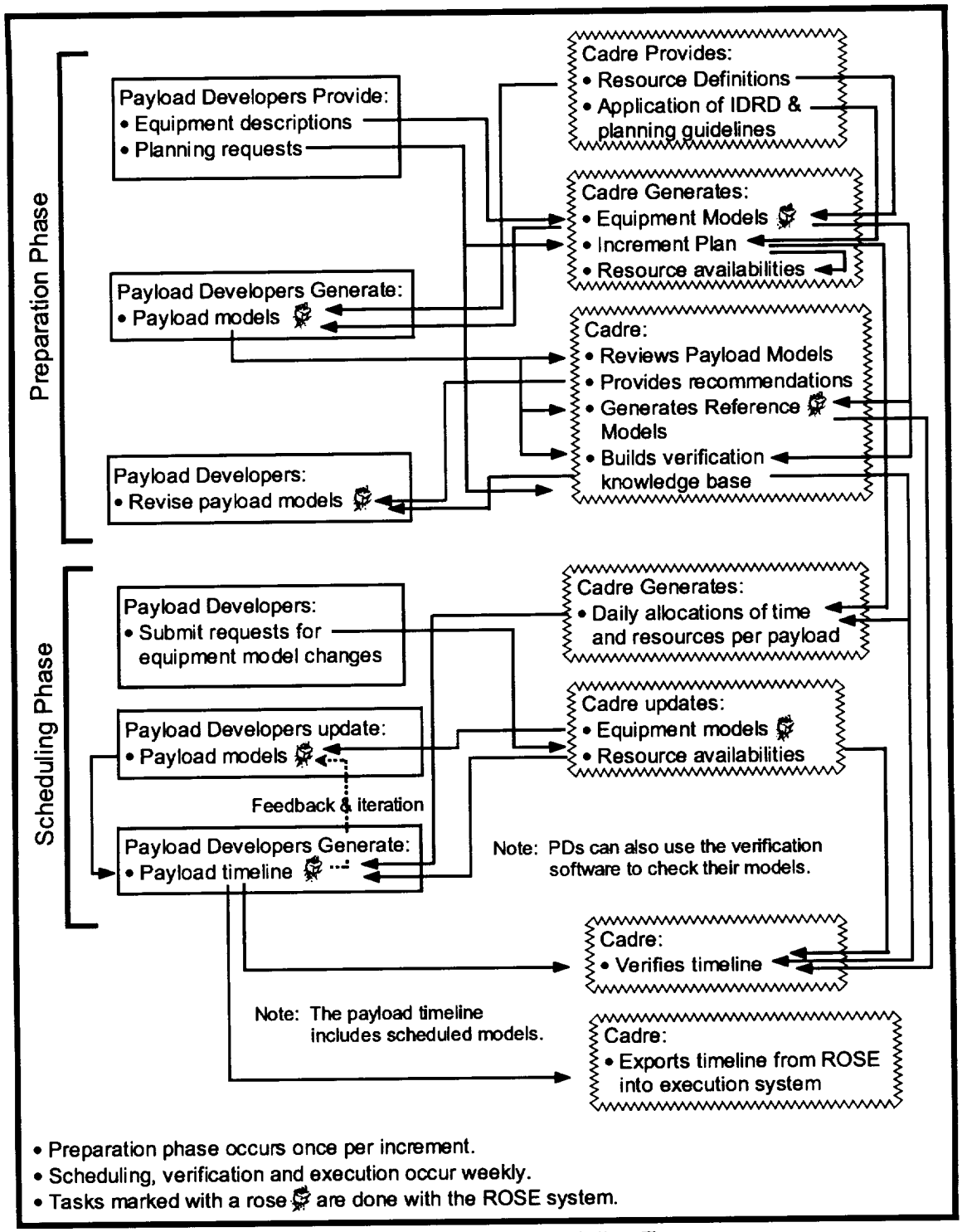

Figure 5. Details of Scheduling Flow

After the equipment models are defined, the payload developer creates the payload models. Normally these models reference only the equipment models and the crew. While these models should be as good as possible, the payload developer will have an opportunity to modify them and add new models during the scheduling phase. The cadre reviews the models and makes comments to the payload developer where appropriate. Based on these preliminary models from the payload developer, the IDRDs, and the payload developer's planning request, the cadre creates a knowledge base that is used to verify the timeline when it is produced. This knowledge base contains a reference

set of models and other information. Based on the feedback from the cadre, the payload developer might update his models.

Scheduling Phase The scheduling phase is divided into a sequence of independent weeks during which the schedule for the-week-after-next is produced. Figure 5 shows the information flow and figure 6 shows the daily activities during this phase. On Monday the payload developer requests updates to the equipment models. The cadre updates the equipment models based on developer requests and the current ISS configuration, generates the daily allocations per payload for the week to be scheduled, and generates the resource availabilities for the entire ROSE installation. The cadre also initializes and activates the instance of ROSE.

On Tuesday and Wednesday, the payload developer generates the schedule for his payload. Generating the schedule is done using the ROSE system to submit a scheduling request (a model) for scheduling. If the model doesn't schedule or schedules at the wrong times, the payload developer is free to change the model and try again – multiple iterations are normally expected. The installation of ROSE will be supporting many simultaneous users who are competing for the shared ISS resources available, but each payload is limited to a payload-specific allocation. When a model is scheduled in ROSE, an abstract of the model that was scheduled is stored with the timeline. Models are never locked against editing; the payload developer can always edit a model and schedule another, but different, occurrence in the timeline. Also on Tuesday and Wednesday, the cadre updates the verification knowledge base.

On Thursday, the cadre verifies the timeline. The cadre uses a comparison tool to compare the scheduled models to the reference models. It is not necessary to verify that the timeline is feasible (doesn't overbook any resources) because the ROSE scheduling engine intrinsically doesn't allow that. When the cadre finds something that violates what they believe is outside of the allowed margins, it is simply deleted from the timeline, and a report is written. This will be viewable by the payload developers.

On Friday, the cadre exports the timeline from ROSE. The model abstracts are also exported; these abstracts are actually models that correspond to the schedule portion of the models that was submitted for scheduling. For example, if a model contained alternate sequences of activities, the abstract would only describe the alternative that was actually scheduled. The exported timeline and models are forwarded to the Payload Operations Integration Function (POIF) for integration with timelines from other partners.

Saturday and Sunday are days off.

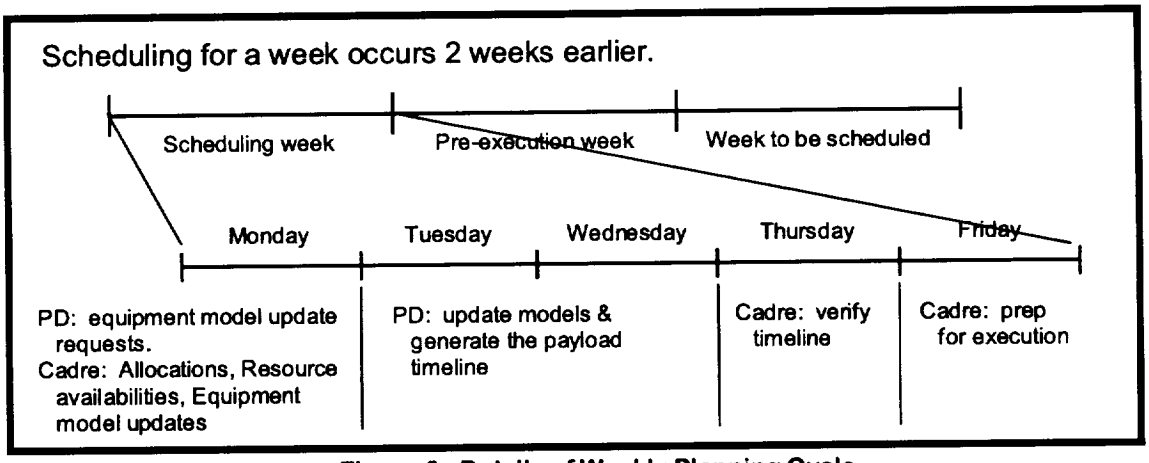

Figure 6. Details of Weekly Planning Cycle

New Software Module The ROSE process requires one new software capability. Model and timeline verification is a new feature of the ROSE-based process. The verification program would primarily compare scheduled models with reference models developed during the preparation phase. While this process accomplishes timeline verification, it only needs to verify the models; the scheduling engine would be trusted to schedule the models correctly.

Issues Rebutted

In discussions with U.S. Payload Control Center personnel, several issues relative to the use of a ROSE-based system have been voiced. This section discusses and rebuts these issues.

The Payload Developers Will "Fudge" The ROSE-based concept includes safeguards to prevent understating requirements. The payload developers do not have the ability to change the resource usage specified for their equipment. They can only understate requirements by shortening the duration or by specifying a different mode. This approach to understating requirements is normally not useful. For example, it does no good to schedule the temperature holding mode of an oven without first scheduling the warm-up mode (which requires more power).

Daily Allocations Are Too-Little / Too-Much What is allocated is adjustable. For most payloads, it will probably suffice to allocate only time and a few scarce resources. Resources for which no allocations are specified would be unlimited. By allocating more or fewer resources, the cadre can exercise increased or decreased control over the timeline. Experience might show that the allocations should provide more or less granularity; i.e., that the proposed daily allocations should specify allocations for a 2-day, or longer, period (less granularity) or for a time-of-day period (more granularity).

Models Are Not Under Configuration Control Configuration control of the models is an implementation feature of the current operations concept. In the ROSE-based concept, the post-scheduling verification of the models and timeline ensures that the invalid models are not scheduled.

The System Is Not Compatible With POIF Software Since ROSE defines and schedules the same ISS resources that other scheduling systems use, this issue refers to the more elaborate models that ROSE supports. However, when the timeline is exported for delivery to the POIF, an abstract of each scheduled model is exported – this abstracted model is compatible. For example: a ROSE model can request crewmember #1 for ten minutes *or* crewmember #2 for twenty minutes and the scheduling engine will make a choice. The exported model would report only the chosen crewmember and corresponding duration.

The System Is Susceptible to Hacking ROSE is a web-based system and the web is inherently un-secure. This is true. But, the security issue is being addressed in the research phase of ROSE and adequate safeguards are being designed into the system. In addition, the information technology itself will provide off-the-shelf remedies (such as virtual private networking) to mitigate some of the problems.

First-Come, First-Serve Approach Is Unfair The ROSE-based concepts uses the increment plan to generate daily allocations of resources on a per-payload basis to solve this problem. Other solutions, such as enabling disadvantaged payloads earlier, are also available.

It's Too Hard For Payload Developers To Use Difficulty of use depends on the quality of the software. Experience and feedback from the modeling methods currently deployed in the iURC program indicates that payloads developers are fully capable of using the graphics specification of requirements methods to describe their payload requirements. The ROSE scheduling engine will be designed to understand and schedule the requirements exactly as modeled. All resource usage (except crew) will be implicitly modeled in the equipment models supplied by the cadre; payload models will only show what equipment and equipment mode is needed. Additionally, the payload developers will have feedback from the schedule engine to reinforce and enhance their modeling skills.

Payload Developers Don't Want To Participate The improvements in the quality of the schedule (from the payload viewpoint) will convince the payload developers that the ROSE-based concept is worth the extra effort required from them.

There Is No Timeline Editor The modeling and scheduling capabilities of ROSE eliminate the need for a timeline editor for use by the payload developers. The execution team will be using a different tool that has a timeline editor.

Benefits

The proposed ROSE-based operations concept will provide both intangible and tangible benefits. The ROSE-based concept allows the payload developers to produce the timeline they want. They are the customers. Giving the customers what they want is the epitome of a better timeline. Compare this to the current concept where a cadre of schedulers produces the timeline with stale models and limited

interaction with payload developers. The scheduling task is distributed to those who best know how to do it, rather than concentrated on a small cadre that is struggling to understand the complex requirements of *all* the payloads.

Better Schedule The quality of the timeline will be significantly improved because those with the most knowledge about the payload and how it should be scheduled produce the timeline. Additionally the timeline will be produced with up-to-date models; the payload developers who know the requirements best, have total control of the models and are free to make updates at any time.

Fewer Change Requests Because the payload developers produce the timeline, they will need to generate far fewer change requests to be implemented by the execution team. Fewer change requests reduces the workload on the execution team and possibly reduces the size of the team.

Less Cadre Time and Burn-Out The current concept is expected to exhaust all but the most dedicated scheduler in a few months. The ROSE-based concept eliminates scheduling as a cadre task. Even though some new tasks are introduced, the number of cadre members can be reduced.

Available to the Crew ROSE is available to the on-board crew. ROSE is a web-based application and the ISS crew will have access to the Internet. The crew could schedule some payload activities; they would act as part of a payload developer's team and schedule payload activities as though they were on the ground.

Operates on Low-Cost Hardware ROSE operates on low-cost computers running the Windows operating system.

Conclusion

The ROSE-based operations concept presented in this paper is an alternative concept that is operationally viable, is low cost, provides more autonomy to payload developers and is presented for future consideration in defining the long term operations concept for ISS payload operations. The ROSE system is based on a proven modeling methodology that adequately and eloquently represents payload requirements. The architecture of the ROSE system has been developed and demonstrated. Research and development on the scheduling engine shows that it is feasible to develop a scheduling engine that matches the modeling methodology. The proposed process can easily be substituted for the current process, with reductions in required personnel. Most significantly, the ROSE-based concept allows the payload developer community to produce the timeline. They are the customers; they know what is required in the timeline better than anyone else; and the ROSE system will provide them with a convenient and robust tool to produce the timeline they want. At the same time, the ROSE-based concept provides adequate controls to ensure that the timeline meets programmatic constraints.

References

Jaap, J.; Davis, E.; and Meyer, P. 1997. Using Common Graphics Paradigms Implemented in a Java Applet to Represent Complex Scheduling. In proceedings of International Workshop on Planning and Scheduling for Space Exploration and Science. 20-1—20-3. Oxnard, California. http://payloads.msfc.nasa.gov/iURC/publications

Online User's Manual for the Interim Users' Requirements Collection System: http://payloads.msfc.nasa.gov/iURC/help

Jaap, J.; and Davis, E. 2000. Can Customers Schedule Their Own Payload Activities? In proceedings of 2nd International Workshop on Planning and Scheduling for Space. San Francisco, California. http://payloads.msfc.nasa.gov/ROSE/publications

Hagopian, J.; and Maxwell, T. 1998. Explicit and Implicit Resources: A Simplified Approach to User Requirements Modeling. In proceeding of the sixth International Symposium on Space Mission Operations and Ground Data Systems (SpaceOps 96). Munich Germany: Spacecraft Operations Oriented International Association (SpaceOps). http://payloads.msfc.nasa.gov/FD40/papers/SimplifiedModeling/3_8.html